IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re application of: Sam IDICULA

Confirmation No.: 3748

Examiner: COLAN, Giovanna B.

Serial No.: 10/648,749

Group Art Unit No.: 2162

Filed on: August 25, 2003

For: IN-PLACE EVALUATION OF XML SCHEMAS

MS Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF

Sir:

This Appeal Brief is submitted in support of the Notice of Appeal filed on October 24, 2008. This is the second Appeal Brief filed in this matter; therefore, the Applicants believe that no fees are required in connection with the filing of this Appeal Brief.

I. REAL PARTY IN INTEREST

Oracle International Corporation is the real party in interest.

II. RELATED APPEALS AND INTERFERENCES

Appellants are unaware of any related appeals or interferences.

III. STATUS OF CLAIMS

Claims 1-12, 14-16, 18-29, and 31-33 have been finally rejected and are the only subjects of this appeal. Claims 13, 17, 30, and 34 were canceled.

IV. STATUS OF AMENDMENTS

The claims were not amended after the Final Office Action.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The present application contains independent Claims 1 and 12, which are summarized below. The claims summarized below are annotated to cross-reference features of the claims to specific examples of those features disclosed in the specification. However, the annotations are not intended to limit the scope of the recited features to those specific examples to which the annotations refer.

Claim 1 recites (with added reference annotations in parenthesis) a method (FIG. 2, method 200) of evolving an Extensible Markup Language (XML) schema (FIG. 1, existing XML schema 106), the method comprising:

receiving (FIG. 2, step 202; paragraphs [0029], [0038], and [0046]), at a schema evolver (FIG. 1, schema evolver 102) that is executing in a computer system (FIG. 3, system 300), a document (FIG. 1, XML document 108) that indicates one or more changes to be made to an existing first XML schema (FIG. 1, existing XML schema 106);

based on said first XML schema and said document, said schema evolver generating (FIG. 2, step 204; paragraphs [0029], [0039], and [0047]) an evolved second XML schema (FIG. 1, evolved XML schema 112); and

based on said second XML schema, generating (FIG. 2, step 206; paragraphs [0030], [0033]-[0035], [0041], and [0048]-[0050]) one or more first Structured Query Language (SQL) statements (FIG. 1, SQL statements 118).

Claim 12 recites (with added reference annotations in parenthesis) a method (FIG. 2, method 200) of generating (FIG. 2, step 206) Structured Query Language (SQL) statements (FIG. 1, SQL statements 118) to alter (paragraphs [0033]-[0035]) database types (FIG. 1, existing database object types 114) in a database system (FIG. 1, database 110) that has definition data that defines (paragraph [0031]) a set of one or more database object types (FIG. 1, existing database objects types 114), the method comprising:

receiving (paragraph [0040]) a first Extensible Markup Language (XML) schema (FIG. 1, XML schema 112); and

based on said first XML schema, generating (FIG. 2, step 206; paragraphs [0030], [0041], and [0048]-[0050]) one or more SQL statements (FIG. 1, SQL statements 118) that, when executed (FIG. 2, step 210; paragraph [0052]), cause a database server (FIG. 1, database server 104) to alter (paragraphs [0033]-[0035]) said set of one or more database object types;

wherein said one or more database object types were generated based on (paragraph [0030]) a second XML schema (FIG. 1, existing XML schema 106) that differs (paragraphs

[0029], [0039], and [0047]) from said first XML schema, wherein said one or more database object types are types of objects within the database system.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Claims 1-3, 6, 18-20, and 23 stand rejected under 35 U.S.C. § 102(e) as being anticipated, allegedly, by U.S. Patent Application Publication No. 2003/0120665 ("Fox").

2. Claims 12, 15-16, 29, 32, and 33 stand rejected under 35 U.S.C. § 102(e) as being anticipated, allegedly, by Fox.

3. Claims 14 and 31 stand rejected under 35 U.S.C. § 102(e) as being anticipated, allegedly, by Fox.

4. Claims 9 and 26 stand rejected under 35 U.S.C. § 102(e) as being anticipated, allegedly, by Fox.

5. Claims 5, 7-8, 22, and 24-25 stand rejected under 35 U.S.C. § 103(a) as being unpatentable, allegedly, over Fox in view of U.S. Patent No. 6,636,845 ("Chau").

6. Claims 10, 11, 27, and 28 stand rejected under 35 U.S.C. § 103(a) as being unpatentable, allegedly, over Fox in view of U.S. Patent Application Publication No. 2002/0007363 ("Vaitzblitz").

7. Claims 4 and 21 stand rejected under 35 U.S.C. § 102(e) as being anticipated, allegedly, by Fox.

VIII. ARGUMENTS

A. The Features of Claims 1-3, 6, 18-20, and 23 Are Not Disclosed, Taught, or Suggested by Fox

According to Claim 1, a schema evolver receives a document that indicates changes that are to be made to a first, existing XML schema. The schema evolver generates a second, evolved XML schema based on the first XML schema and the document that indicates the changes. Thus, Claim 1 is all about generating a new, evolved ("second") XML schema, and has nothing to do with transforming documents that conform to a schema into documents that conform to a different schema.

In contrast, Fox is all about transforming documents that conform to a schema into documents that conform to a different schema, and has nothing to do with generating a new, evolved XML schema.

Significantly, Fox doesn't generate the second schema based on the first schema and the "transformation." Instead, Fox generates the "transformation" based on the first schema (the "source data schema") **and** the second schema (the "target data schema"). Because the transformation is generated based on the second schema, the "transformation" clearly cannot be generated until the second schema already exists. Since the transformation cannot be generated until the second schema already exists, it make no sense to say that the second schema is generated based on the transformation.

Fox describes how this "transformation" is generated as follows: "At step 120, a source data schema and a target data schema are imported" (paragraph [0104]). "At step 180, a transformation is derived for transforming **data conforming with** the source data schema into **data conforming with** the target data schema" (paragraph [0107]).

It is clear from this description that the "transformation," which is "for transforming data that conforms to the source data schema into data that conforms to the target data schema" is "derived" from the source **and target** data schemas rather than the target data schema being derived from the source data schema and the "transformation."

The errors of fact underlying the Examiner's position have been set forth. There are, additionally, numerous places in Fox that are starkly inconsistent with the Examiner's interpretation. For example, Fox shows that schema receiver 210 and transformation generator 260 **receive, as input, both a source data schema and a target data schema**. Based on this input, transformation generator 260 **outputs a "derived transformation."**

For another example, paragraph [0050] says that there is "a need for a tool that can transform data **conforming to** a first schema into data **that conforms to** a second schema." Paragraph [0049] discusses the problem of different companies using different **existing** schemas, and how this problem makes it difficult for these companies to use each other's data (because the data conform to different **existing** schemas).

For another example, paragraph [0051] says that the "present invention provides a method and system **for deriving transformations** for transforming data from one schema to another." It is the **transformation** that is derived based on the **schemas** rather than a **schema** being derived based on the **transformation**. Paragraph [0051] also mentions that an XSLT script may be generated, and explains that the XSLT script can be applied to (a) documents that **conform to** the source XML schema in order to generate (b) documents that **conform to** the target XML schema. Clearly, the XSLT script transforms the documents that **conform to** the XML schemas rather than the XML schemas **themselves**.

For another example, paragraph [0061] says, "**Given a source XML schema and a target XML schema** . . . an appropriate **transformation** of source to target XML documents **is generated**." Clearly, the **transformation** is generated based on the **source and target XML schemas** rather than the **target XML schema** being generated based on the **transformation**.

Thus, Fox is virtually brimming with statements that support the Applicants' position and undermine the Examiner's position. Fox **does not** disclose, teach, or suggest that a schema evolver generates an evolved XML schema based on both (a) an existing XML schema and (b) a document that indicates changes that are to be made to the existing XML schema, as required by Claim 1.

One advantage of the method of Claim 1 is that a user does not need to manually change a first schema into a second schema. Instead, for example, the user can generate the document that indicates the changes, and then the schema evolver can use the document to make the changes to the first schema in order to produce the second schema automatically. Because Fox's approach requires the source and target schemas to exist before the "transformation" (alleged "document") is derived, Fox's approach does not confer such an advantage.

In reply to the above arguments, the Examiner argues, on Page 13 of the Final Office Action, that "the features upon which the applicant relies (i.e., an 'evolved XML schema' and 'an existing XML schema') are not recited in the rejected claims. This is utterly false. Claim 1 clearly recites both "an existing first XML schema" and "an evolved second XML schema."

The Examiner further argues, on Page 13 of the Final Office Action, that USPTO personnel are to give claims their broadest reasonable interpretation in light of the supporting disclosure. The Appellants respond that the Examiner's interpretation of the claims is completely unreasonable and does not appear to be based at all upon the Appellants' description in the specification. Indeed, the Examiner's interpretation of the claims appears to totally ignore the description in the specification in order to support the Examiner's unreasonable interpretation of the claims.

On Page 14 of the Final Office Action, the Examiner points to an example in Fox in which an XSLT script is generated based on both an first **existing** XML schema (the SwissAir XML schema) and a second **existing** XML schema (the British Airways XML schema). However, even the portions of Fox that the Examiner cites indicate that this generated XSLT script is for transforming **documents that conform**to the first XML schema into **documents that conform** to the second XML schema (rather than transforming the first XML schema into the second XML schema). On Page 15 of the Final Office Action, the Examiner once again mischaracterizes this passage in Fox as meaning that the second XML schema is generated based on the first XML schema and the XSLT script. This mischaracterization completely ignores the very passage of Fox that the Examiner just quoted: the passage that expressly states that XSLT script is generated based on both the SwissAir and the British Airways XML schemas. Clearly, the XSLT document cannot be used to generate the British Airways XML schema, since the British Airways XML schema must exist initially in order to generate the XSLT document in the first place. The Applicants admit that the XSLT document, once generated, can then be used to transform XML documents **that conform to** the SwissAir XML schema into XML documents **that**

conform to the British Airways XML schema, but this is not the same as generating the British Airways XML schema **itself**. The difference between (a) transforming an XML schema and (b) transforming a document **that conforms to** an XML schema (but not the XML schema itself) appears to be a difference that the Examiner either does not understand or is deliberately ignoring.

The Examiner then responds, on Page 15 of the Final Office Action, that "the Examiner interprets that generating a transformation of a second schema implies generating a second schema as claimed." However, the portions of Fox that the Examiner cited immediately before this "clarification" **do not discuss** "generating a transformation **of a second schema**." Instead, the portions of Fox that the Examiner cited discuss generating a transformation of a SwissAir XML document (an XML document **that conforms to** the SwissAir XML schema, not the SwissAir XML schema itself) into a British Airways XML document (an XML document **that conforms to** the British Airways XML schema, not the British Airways XML schema itself). Fox does not disclose "generating a transformation **of a second schema**" as the Examiner asserts, and so the Examiner should not read Fox as implying the generation of a second schema. As the Appellants have already shown, the XSLT script that is used to transform the SwissAir XML document into the British Airways XML document is generated based on two XML schemas that already exist: the SwissAir XML schema and the British Airways XML schema.

In a desperate last attempt to muddy the waters, the Examiner, on Page 15, cites a portion of paragraph [0482] of Fox, which discusses inserting "code" within a "bloc," where the insertion of this "code" may include "the creation of new templates if needed." The Appellants guess that the Examiner is hoping that the Honorable Board will be confused and

believe that the word "template" in this portion of Fox somehow refers to an XML schema, since an XML schema can be thought of, in one way, as a template for the XML documents that conform to that XML schema. However, when viewed in context of the text surrounding the quoted passage, it becomes clear that the "creation of templates" referred to in paragraph [0482] has **nothing to do** with an XML schema. The first sentence of paragraph [0482] says: "Preferably, the general rule **for producing XSLT script** associated with a target choice bloc is as follows." Clearly, the creation of "templates" refers to the production of XSLT script (the generated "transformation") rather than the modification or generation of an XML schema. The last sentence of paragraph [0482] actually refers to inserting code "into the XSLT" (not into an XML schema). The following two paragraphs, [0483] and [0484], discuss an example of a portion of an XSLT document (in paragraph [0484]) that would be generated in response to the existence of certain text in **the target schema** (in paragraph [0483]). Two things are again apparent from these paragraphs: (1) the XSLT document, not an XML schema, is being generated, and (2) **the target schema** exists **already** and is consulted in order to generate the XSLT document. The Examiner has not helped her position by citing text at this portion of Fox. The references to "templates" in paragraph [0482] do **not** refer to XML schemas, as the Examiner apparently hopes.

For at least the above reasons, Claim 1 is patentable over Fox under 35 U.S.C. § 102(e). By virtue of their dependence from Claim 1, Claims 2-3, 6, 18-20, and 23 include the features of Claim 1 distinguished from Fox above. As a result, Claims 1-3, 6, 18-20, and 23 are patentable over Fox under 35 U.S.C. § 102(e). The rejection of Claims 2-3, 6, 18-20, and 23 should be reversed.

B. The Features of Claims 12, 15, 16, 29, 32 and 33 Are Not Disclosed, Taught, or Suggested by Fox

Among other features, Claim 12 recites "wherein said one or more database object types were generated based on a second XML schema that differs from said first XML schema."

The Examiner alleges that Fox discloses this limitation in paragraph [0453], lines 8-11, which say: "For example, if the given table column has data type VARCHAR2, then the choice of properties may only include properties with target type string, or compositions of properties whereby the final property in the composition has target type string."

This paragraph of Fox, and the surrounding paragraphs, do not refer to the generation of **database object types**. As can be seen by reading the preceding paragraphs [0450] through [0452], this paragraph of Fox is actually discussing the **mapping** of database tables to classes (in an "ontology model"), and the mapping of fields in those database table to properties of those classes. Paragraph [0453] discusses how an existing type of a database column, VARCHAR2, can be mapped to a conforming field type such as "string" (since columns that are of type VARCHAR2 usually contain textual data that can be assigned to a string type). Although "string" clearly is a type, it is **not a database type**; in paragraph [0435], "string" is the type of a field in a class in the "ontology model," rather than a database object type. Although components of the "ontology model" may be **mapped** to various components in a relational database, the decision that a particular component of the "ontology model" should be a particular type **does not** involve the generation of a **database object type** in any way. No generation of a database object type can be inferred from paragraph [0453] or the surrounding text.

Even if VARCHAR2 is considered to be a database object type, Fox does not generate this type. Fox only maps a column having this type to a field in an ontology model, and gives that field an appropriate corresponding (non-database) type. The cited portions of Fox do not refer to the generation of database object types based on an XML schema. Instead, Fox refers to the mapping of existing database components, which have types, to ontology model components. In order for Fox to do this, the database components, with their types, must already exist.

On Page 17 of the Final Office Action, the Examiner alleges that Fox's paragraph [0455] discusses the generation of one or more database object types. This is not the case, however. Paragraph [0455] discusses an instance of a "class," called "person." "Person" has properties such as "name" and "ID number." These are merely the properties of a class in an "ontology model." These are not database object types. Paragraph [0455] does not say **anything** about the generation of database object types, let alone the generation of database object types based on an XML schema.

Additionally, on Page 17 of the Final Office Action, the Examiner argues that a "template" corresponds to a "database object type." The Appellants are unaware of any database object type known as "template." The Examiner's bare assertion that a "template" is a "database object type" appears to be wholly without evidentiary support.

Therefore, Fox does not disclose, teach, or suggest "wherein said one or more database object types were generated based on a second XML schema that differs from said first XML schema" as recited in Claim 12.

By virtue of their dependence from Claim 12, Claims 14-16, 29, 32, and 33 include the features of Claim 12 distinguished from Fox above. As a result, Claims 12, 14-16, 29,

32, and 33 are patentable over Fox under 35 U.S.C. § 102(e). The rejection of Claims 12, 14-16, 29, 32, and 33 should be reversed.

C. The Features of Claims 14 and 31 Are Not Disclosed, Taught, or Suggested by Fox

Claim 14 depends from Claim 12 and further recites, "wherein said first XML schema was generated based on said second XML schema."

The Examiner alleges that Fox discloses this feature of Claim 14 in paragraph [0502], which reads, in its entirety:

> Preferably, components are implemented as objects that can send and receive Messages to and from other objects. Thus, for example, an indirect property P_2OP_1 and an indirect inheritance (C, D) are implemented as their own objects. Direct dependencies among the objects are indicated by in FIG. 27 by directed edges within the dependency graph. If a first object depends on a second object either through a direct (single edge) or indirect (multiple edge) dependency, then modification or deletion of the second object potentially impacts the first object. For example, referring to FIG. 27, a constraint depends directly on an indirect property, and a mapping depends indirectly on an indirect property.

Although the above paragraph refers to dependencies between **objects**, the above paragraph does not indicate that these objects are different **XML schemas** or that these objects were generated **based on different XML schemas**. There appears to be absolutely no relation or similarity between the **objects** described in this paragraph and the **XML schemas** recited in Claim 14. Appellants concede that is it well known that one object may depend upon another object. However, it does not follow from this well-known fact that an evolved XML schema must be generated based on an existing XML schema.

As is discussed above with reference to Claim 1, Fox does not disclose, teach, or suggest generating an evolved XML schema based on an existing XML schema. Instead, Fox is concerned with transforming (a) documents that **conform** to one schema into (b)

documents that **conform** to another schema. Fox's approach assumes that the two schemas already exist, and does not propose any approaches for generating either schema.

Therefore, Fox does not disclose, teach, or suggest "wherein said first XML schema was generated based on said second XML schema" as recited in Claim 14.

By virtue of its dependence from Claim 14, Claim 31 includes the features of Claim 14 distinguished from Fox above. As a result, Claims 14 and 31 are patentable over Fox under 35 U.S.C. § 102(e). The rejection of Claims 14 and 31 should be reversed.

D. The Features of Claims 9 and 26 Are Not Disclosed, Taught, or Suggested by Fox

Claim 9 depends from Claim 1 and further recites, "wherein said one or more changes are expressed as one or more instances of **one or more XML types specified by a third XML schema**." For example, in paragraphs [0055]-[0058] of the present application, three XML types, "append-node," "insert-node-before," and "delete-node," are described. These XML types are specified in an "xdiff schema" which is separate from both the existing schema (e.g., existing XML schema 106) and the evolved schema (e.g., evolved XML schema 112). The XML document (e.g., XML document 108), which describes the changes that are to be made to the existing schema in order to produce the evolved schema, describes those changes in terms of XML change elements. Each of these XML change elements is an instance of an XML type (e.g., "append-node," "insert-node-before," or "delete-node") that is specified in the "xdiff" ("third") XML schema.

The Examiner alleges that Fox discloses this feature of Claim 9 in paragraph [0200], which reads, in its entirety:

> Reference is now made to FIGS. 11A-11R, which are illustrations of a for transforming data from one XML schema to another using the Coherence

> software application, in accordance with a preferred embodiment of the present invention. Shown in FIG. 11A is a window with package view of an Airline Integration ontology model in its left lane. The left pane displays classes from a fundamental package. A class Date is shown highlighted; and its properties are shown in the right pane. Fundamental packages are used for standard data types. Shown in FIG. 11B is a window with a hierarchical view of the Airline Integration ontology model in its left pane. The left pane indicates that FrequentFlyer is a subclass of Passenger, Passenger is a subclass of Person, and Person is a subclass of Being. The right pane displays general information about the class FrequentFlyer.

The above paragraph does **not** disclose, teach, or suggest that the changes that are to be made to one XML schema (e.g., existing XML schema 106) are expressed as instances of XML types (e.g., "append-node," "insert-node-before," and "delete-node") that are specified by another XML schema (e.g., the "xdiff schema").

Therefore, Fox does not disclose, teach, or suggest "wherein said one or more changes are expressed as one or more instances of **one or more XML types specified by a third XML schema**" as recited in Claim 9.

By virtue of its dependence from Claim 9, Claim 26 includes the features of Claim 9 distinguished from Fox above. As a result, Claims 9 and 26 are patentable over Fox under 35 U.S.C. § 102(e). The rejection of Claims 9 and 26 should be reversed.

E. The Features of Claims 5, 7-8, 22, and 24-25 Are Not Disclosed, Taught, or Suggested by Fox or Chau

By virtue of their dependence from Claim 1, Claims 5, 7-8, 22, and 24-25 inherit the features of Claim 1 that are distinguished from Fox above. Therefore, Fox, taken individually, does not disclose, teach, or suggest the subject matter of any of Claims 5, 7-8, 22, and 24-25.

Chau also does not disclose, teach, or suggest the distinguished features of Claim 1 that are inherited by Claims 5, 7-8, 22, and 24-25. Indeed, the Examiner does not even allege that Chau discloses, teaches, or suggests these inherited features.

Consequently, even if Fox and Chau could be combined, the combination would still fail to disclose, teach, or suggest that a schema evolver generates an evolved XML schema based on both (a) an existing XML schema and (b) a document that indicates changes that are to be made to the existing XML schema, as required by each of Claims 5, 7-8, 22, and 24-25.

As a result, Claims 5, 7-8, 22, and 24-25 are patentable over Fox and Chau under 35 U.S.C. § 103(a). The rejection of Claims 5, 7-8, 22, and 24-25 should be reversed for at least the reasons discussed above in connection with Claim 1.

F. The Features of Claims 10, 11, 27, and 28 Are Not Disclosed, Taught, or Suggested by Fox or Vaitzblitz

By virtue of their dependence from Claim 1, Claims 10, 11, 27, and 28 inherit the features of Claim 1 that are distinguished from Fox above. Therefore, Fox, taken individually, does not disclose, teach, or suggest the subject matter of any of Claims 10, 11, 27, and 28.

Vaitzblitz also does not disclose, teach, or suggest the distinguished features of Claim 1 that are inherited by Claims 10, 11, 27, and 28. Indeed, the Examiner does not even allege that Vaitzblitz discloses, teaches, or suggests these inherited features.

Consequently, even if Fox and Vaitzblitz could be combined, the combination would still fail to disclose, teach, or suggest that a schema evolver generates an evolved XML

schema based on both (a) an existing XML schema and (b) a document that indicates changes that are to be made to the existing XML schema, as required by each of Claims 10, 11, 27, and 28.

As a result, Claims 10, 11, 27, and 28 are patentable over Fox and Vaitzblitz under 35 U.S.C. § 103(a). The rejection of Claims 10, 11, 27, and 28 should be reversed for at least the reasons discussed above in connection with Claim 1.

G. The Features of Claims 4 and 21 Are Not Disclosed, Taught, or Suggested by Fox

Claim 4 recites "wherein said first SQL statements, when executed, cause one or more database object types to be deleted." The Examiner merely states that the deletion of an object implies the deletion of an object type. This is simply untrue. An object may be deleted without ever deleting that object's type. For example, if a class "employee" is created, and if an instance of that class, "Bob" is created, then "Bob" is an object, and "employee" is Bob's type. Other objects of the same class, such as "Fred" and "Jim," also may be instantiated, and these objects also would have the type "employee." Subsequent deletion of the "Bob" object **would not** automatically cause the "employee" class also to be deleted. Indeed, because other instances of that class, namely "Fred" and "Jim," would still exist, the "employee" class must continue to exist despite the deletion of "Bob."

Similarly, in the context of database systems, the deletion of a database object that has a certain database object type does **not** imply that the database object type is also deleted. Clearly, if one deletes a database object that is of type "VARCHAR2," then the "VARCHAR2" type will continue to exist—other database objects might also have this type,

and even if no other database objects have this type, later created database objects might be assigned this type.

There simply is no justification for the unsupported implication on which the Examiner's rejection of Claims 4 and 21 is based. The rejections of Claims 4 and 21 should be reversed.

IX. CONCLUSION AND PRAYER FOR RELIEF

Based on the foregoing, it is respectfully submitted that the rejections of Claims 1-12, 14-16, 18-29, and 31-33 lack the requisite factual and legal bases. Appellants respectfully request that the Honorable Board **reverse** the rejections of Claims 1-12, 14-16, 18-29, and 31-33.

Respectfully submitted,

HICKMAN PALERMO TRUONG & BECKER LLP

Date: December 24, 2008

Christian A. Nicholes
Registration No. 50,266

2055 Gateway Place, Suite 550
San Jose, California 95110-1089
Tel: (408) 414-1224
Fax: (408) 414-1076

CLAIMS APPENDIX

1. A method of evolving an Extensible Markup Language (XML) Schema, the method comprising:

   receiving, at a schema evolver that is executing in a computer system, a document that indicates one or more changes to be made to an existing first XML schema;

   based on said first XML schema and said document, said schema evolver generating an evolved second XML schema; and

   based on said second XML schema, generating one or more first Structured Query Language (SQL) statements.

2. The method of Claim 1, wherein said first SQL statements, when executed, cause one or more database object types to be created.

3. The method of Claim 1, wherein said first SQL statements, when executed, cause one or more database object tables to be created.

4. The method of Claim 1, wherein said first SQL statements, when executed, cause one or more database object types to be deleted.

5. The method of Claim 1, wherein said first SQL statements, when executed, cause one or more database object tables to be deleted.

6. The method of Claim 1, wherein said first SQL statements, when executed, cause one or more database object types to be altered.

7. The method of Claim 1, wherein said first SQL statements, when executed, cause one or more database object tables to be altered.

8. The method of Claim 1, wherein said first SQL statements, when executed, cause one or more database object instances to be altered.

9. The method of Claim 1, wherein said one or more changes are expressed as one or more instances of one or more XML types specified by a third XML schema that is separate from said first XML schema and said second XML schema.

10. The method of Claim 1, further comprising:
generating one or more second SQL statements that, when executed, cause effects of said one or more first SQL statements to be reversed.

11. The method of Claim 10, further comprising:
determining, while executing said one or more first SQL statements, whether an error has occurred; and
in response to determining that an error has occurred, executing one or more of said one or more second SQL statements that, when executed, cause effects of said one or more first SQL statements that have been executed to be reversed.

12. A method of generating Structured Query Language (SQL) statements to alter database types in a database system that has definition data that defines a set of one or more database object types, the method comprising:
receiving a first Extensible Markup Language (XML) schema; and
based on said first XML schema, generating one or more SQL statements that, when executed, cause a database server to alter said set of one or more database object types;
wherein said one or more database object types were generated based on a second XML schema that differs from said first XML schema, wherein said one or more database object types are types of objects within the database system.

13. (Canceled)

14. The method of Claim 12, wherein said first XML schema was generated based on said second XML schema, wherein said second XML schema exists prior to said first XML schema, wherein said first XML schema is evolved from said second XML

schema, and wherein said first XML schema and said second XML schema are different XML schemas.

15. The method of Claim 12, wherein said one or more SQL statements, when executed, cause said database server to create one or more of said one or more database object types.

16. The method of Claim 12, wherein said one or more SQL statements, when executed, cause said database server to delete one or more of said one or more database object types.

17. (Canceled)

18. A volatile or non-volatile computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 1.

19. A volatile or non-volatile computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 2.

20. A volatile or non-volatile computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 3.

21. A volatile or non-volatile computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 4.

22. A volatile or non-volatile computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 5.

23. A volatile or non-volatile computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 6.

24. A volatile or non-volatile computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 7.

25. A volatile or non-volatile computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 8.

26. A volatile or non-volatile computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 9.

27. A volatile or non-volatile computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 10.

28. A volatile or non-volatile computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 11.

29. A volatile or non-volatile computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 12.

30. (Canceled)

31. A volatile or non-volatile computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 14.

32. A volatile or non-volatile computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 15.

33. A volatile or non-volatile computer-readable storage medium carrying one or more sequences of instructions which, when executed by one or more processors, causes the one or more processors to perform the method recited in Claim 16.

34. (Canceled)

EVIDENCE APPENDIX

None.

RELATED PROCEEDINGS APPENDIX

None.